Insecurity of MuSig and Bellare-Neven Multi-Signatures with Delayed Message Selection

Sela Navot

University of Washington, Seattle, USA

Abstract. Multi-signature schemes in pairing-free settings require multiple communication rounds, prompting efforts to reduce the number of signing rounds that need to be executed after the signers receive the message to sign. In MuSig and Bellare-Neven multi-signatures, the signing protocol does not use the message until the third (and final) signing round. This structure seemingly allows pre-processing of the first two signing rounds before the signers receive the message. However, we demonstrate that this approach compromises security and enables a polynomial time attack, which uses the algorithm of Benhamouda et al. to solve the ROS problem.

Keywords: multi-signatures · ROS problem

1 Introduction

Multi-signature schemes [IN83] allow a group of signers to provide a succinct joint signature for an agreed upon message. However, pairing-free discrete-log based multi-signature schemes require multiple rounds of communications to sign a message. To make them closer to non-interactive schemes, some schemes allow all but one of the communication rounds to be completed before the message to be signed or the identity of all the signers have been determined. Prominent examples include the two round scheme MuSig2 [NRS21], the similar DWMS [AB21], and consequent variants [CKM21, BTT22, TZ23], which are proved secure when the first round is pre-processed before the selection of the message and the signing group.

In this paper, we take another look at the classical scheme of Bellare and Neven [BN06] (which we refer to as Bellare-Neven multi-signatures or BN) and MuSig [MPSW19], which are practical *three* round multi-signature schemes. In both schemes, the message to be signed is not used by the signers until the third signing round, and hence they can be executed correctly when the first two rounds are pre-processed before the message is determined. This raises the question of whether these two schemes are secure in such a setting.

Prior literature contains some ambiguities regarding the possibility of delayed message selection for BN and MuSig. The BN multi-signatures description [BN06] explicitly states that the message is selected before the first signing round, but there is no justification as to why this is the case. Furthermore, the MuSig paper [MPSW19] does not explicitly state at which round the message needs to be determined.[1] A concurrent and a later paper proving the security of MuSig [BDN18, BD21] present a more detailed description of the scheme in which the message is explicitly selected before the first signing round, but there is no justification for why early message selection is needed.

[1] It does mention that the first two signing round of MuSig are identical to those of BN multi-signatures, which implies implicitly that the message is selected before the first signing round, but this is never made explicit in the scheme description.

Date of this document: 2024-07-04.

This paper shows that delayed message selection in BN and MuSig leads to a vulnerability by presenting practical polynomial time attacks. We emphasize that the security of MuSig and BN is not compromised and nothing wrong is proved in the corresponding papers. However, we use this paper to advise against a simple optimization that renders the schemes insecure, and, more generally, highlight the value of precise protocol descriptions.

Sub-exponential attacks. In a blog post [Nic19], Jonas Nick shows that if the first two signing rounds are pre-processed before the message is selected, then MuSig is vulnerable to an attack in sub-exponential time using Wagner's algorithm for the generalized birthday problem [Wag02]. As a first step, we present this attack and expand it to work against BN multi-signatures when used with pre-processing of the first two signing rounds.

In these attacks, the adversary initiates multiple signing sessions concurrently (say ℓ sessions) and observes the execution of the first two signing rounds. Then, the adversary chooses the message to be signed in each of the signing sessions, completes the session, and uses the resulting multi-signatures to forge a signature on a message of their choice. In the attack against MuSig the adversary only needs to control the execution of the protocol and pick the messages to be signed in the ℓ legitimate sessions. For our attacks against BN multi-signatures, the adversary also needs different signers to complete a signing session for different messages, which is possible when the adversary mediates signers communication or when they collude with all but one of the signers.[2] The runtime of these attack is dominated by the runtime of Wagner's algorithm, which is more efficient for larger values of ℓ.

Polynomial time attacks. Then, we present polynomial time attacks against BN and MuSig, which completely break their existential unforgeablity when used with delayed message selection. The polynomial attacks use the ideas from the sub-exponential attacks and the algorithm of Benhamouda et al. [BLL+21] to solve the ROS problem [Sch01].

The polynomial time attacks work in the same setting as the sub-exponential attacks, requiring the adversary to control the execution of the protocol, and in the case of BN multi-signatures also mediate the communication of the signers or corrupt all but one of them. As with the sub-exponential attack, the adversary first initiates ℓ signing sessions concurrently, only now we require that $\ell \geq \lceil \log(p) \rceil$ (where p is the order of the underlying group). Then, they choose messages for which to complete the ℓ sessions and use the results to construct a signature for a message of their choice. However, while the ℓ messages for which the adversary obtains legitimate signatures must be selected by the adversary after the second signing round, they can be selected from a set of two arbitrarily chosen messages, allowing the adversary to pick messages that the honest signers are willing to sign.

This paper. Bellare and Dai [BD21] claim that much of the ambiguity and security issues found in multi-signatures stem from lack of detail in the syntax and security definitions. Their claim applies to the ambiguity that we address in this paper, regarding at which round the message to sign is selected in MuSig. Hence, we begin with detailed preliminaries including a syntax and an unforgeability definition for multi-signature schemes, as well as a pseudocode description of the secure and the insecure versions of MuSig. Next, we present our attacks, which are our main contribution in this paper. The attacks section is intended to stand alone and can be read without the definitional sections.

[2] This is possible if the adversary is one of the signers in a group of two signers, for example.

2 Preliminaries

2.1 Notation.

For a positive integer n, we use $\mathbb{Z}_n$ to refer to the ring of integers modulo n equipped with modular addition and multiplication. Addition and subtraction of $\mathbb{Z}_p$ elements are modular, unless otherwise stated. We use multiplicative notation for all other groups. Logarithms are to the base 2.

In our pseudocode, we use $\leftarrow$ to denote assignment and use $\leftarrow\!\!\$$ for randomized assignment. In particular, we write $x \leftarrow\!\!\$ S$ to denote assigning a uniformly random element of a finite set S to x and $x \leftarrow\!\!\$ R(x_1, \dots)$ to denote executing a randomized algorithm R with input $x_1, \dots$ and a uniformly random coin and assigning the output to x. We use subscripts for array indexing and $\perp$ to denote an error value. All variables are assumed to be uninitialized until assigned a value.

2.2 Multi-Signatures Specifications

Multi-signature schemes allow groups of signers to provide a succinct joint signature for an agreed upon message. More specifically, a valid multi-signature by a group of n signers proves that each of the n signers have participated in the signing protocol in order to sign the corresponding message with this group of signers.

Multi-signature schemes are expected to be unforgeable in the plain public key model [BN06], which denotes the setting where each signer has a public key and is not required to prove ownership of an associated secret key nor participate in a distributed key generation protocol. This allows signers to use the same public key to sign multi-signatures with different groups.

Key aggregation. A multi-signature scheme supports key aggregation if a multi-signature can be verified with respect to a single short *aggregate key* of the group, as opposed to the complete list of the public keys of all signers. This property is achieved by MuSig [MPSW19] (though not by BN multi-signatures [BN06]), where the resulting multi-signature is an ordinary Schnorr signature [Sch90] with respect to the aggregate key of the group.

Syntax and correctness. A multi-signature scheme MS is defined by the following algorithms.

Key generation: The algorithm $\mathsf{MS.Kg}$ is used for key generation, and is run individually by each signer. It takes no input (apart from the public parameters of the scheme) and outputs a secret-public key pair.

Signing: The collection $(\mathsf{MS.Sign}_r)_{r=1}^{\mathsf{MS.nr}}$ specifies the procedures for each signing round that is run by each signing party individually, where $\mathsf{MS.nr}$ is the number of signing rounds that is specified by the scheme. The input for each signing round may include the message to sign, a vector of public keys, or the output of previous signing rounds. The multi-signature is the output of the last signing round.

A scheme also specifies the last interactive signing round $\mathsf{MS.lir}$, after which it is possible to construct a multi-signature without knowledge of the signers secret information (it is the last round that needs to be completed by all signers).

Key aggregation: If MS supports key aggregation, the algorithm $\mathsf{MS.KeyAgg}$ takes a list of n public keys $(\mathsf{pk}_i)_{i=1}^n$ as input and outputs a single aggregate verification key $\widetilde{\mathsf{pk}}$.

Verification: If MS does not support key aggregation, it has an algorithm $\mathsf{MS.Verify}$ that takes a list of public keys, a signature, and a message as input and outputs whether the signature is valid. If MS supports key aggregation, it has the algorithm $\mathsf{MS.AggVer}$ with the same functionality that takes an aggregated public key instead of a list of public keys.

Note that if a scheme supports key aggregation, then a standard Verify algorithm can be obtained by executing $\mathsf{MS.AggVer}(\mathsf{MS.KeyAgg}((\mathsf{pk}_i)_{i=1}^n), m, \sigma)$. Hence, we use $\mathsf{MS.Verify}$ in the security definition without loss of generality.

Using the convention of [BD21], each signer i maintains a state $i.\mathsf{st}$ which includes their long-standing private and public keys $i.\mathsf{st.sk}$ and $i.\mathsf{st.pk}$. Additionally, they maintain information for each signing execution s, denoted by $i.\mathsf{st}_s$. This includes $i.\mathsf{st}_s.n$, $(\mathsf{st}_s.\mathsf{pk}_j)_{j=1}^{i.\mathsf{st}_s.n}$, $i.\mathsf{st}_s.m$, $i.\mathsf{st}_s.\mathsf{rnd}$, and $i.\mathsf{st}_s.\mathrm{me}$, which refers to the number of signers in the group, the public keys of the signers, the message to sign, the current execution round, and the index of signer i within the signing group. The session state may also include other information, such as the discrete log of a nonce. To avoid rewinding attacks, it is required that a signer refuses requests to run Sign_ℓ for a session s if $\ell \neq i.\mathsf{st}_s.\mathsf{rnd} + 1$.

In an honest execution of a multi-signature scheme, each party runs the key generation algorithm independently. To sign a message, the signing rounds are executed in sequential order by each signer, with the output of each signing round from all participating signers often used as part of the input for consequent signing rounds. The multi-signature is the output of the last signing round, and correctness requires that an honest execution of the signing protocol results in a valid signature.

2.3 Security of Multi-Signatures

Games framework. We use a simplification of the game based framework of [BR06] for our security definition. The security definition is described as a game, with Init and Fin procedures, and as well as some other oracles. When an adversary $\mathcal{A}$ plays a game, Init is executed and its output is given as the input to $\mathcal{A}$. Now $\mathcal{A}$ is executed, and it may call the oracles with adaptively chosen inputs. When $\mathcal{A}$ terminates, Fin is executed with the output of $\mathcal{A}$ as its input, and returns the output of the game. The advantage of $\mathcal{A}$ is the probability that this output is true.

Given a game-based security definition, a scheme is secure if no polynomial time adversary can achieve non-negligible advantage, where "polynomial" and "negligible" are defined in terms of the security parameter of the scheme. In this paper the security parameter is always $\log(p)$, where p is the order of the underlying group used by the scheme.

Security definition. Multi-signatures are expected to satisfy Existential Unforgeability Against Chosen Message Attacks (MS-EUF-CMA), referring to security against an adversary who can query a signer for signatures on messages of their choice with arbitrary signing groups. In the corresponding game, there is a signer that the adversary attempts to compromise. The adversary can interact with the honest signer via a signing oracle by providing inputs of the adversary's choice, including choosing the message to be signed and the group of signers to sign the message with, and the signing sessions can happen concurrently. To win, the adversary outputs a non-trivial valid multi-signature for a group that includes the honest signer, where non-trivial means that it is valid for a message and signing group for which and with whom the honest signer did not participate in the signing protocol.

There is a subtlety regarding at which signing round is a forgery trivial. Some authors (including those proving the security of MuSig [MPSW19, BDN18, BD21] and BN [BD21]) consider a forgery trivial if a signing session *has begun* with the corresponding message

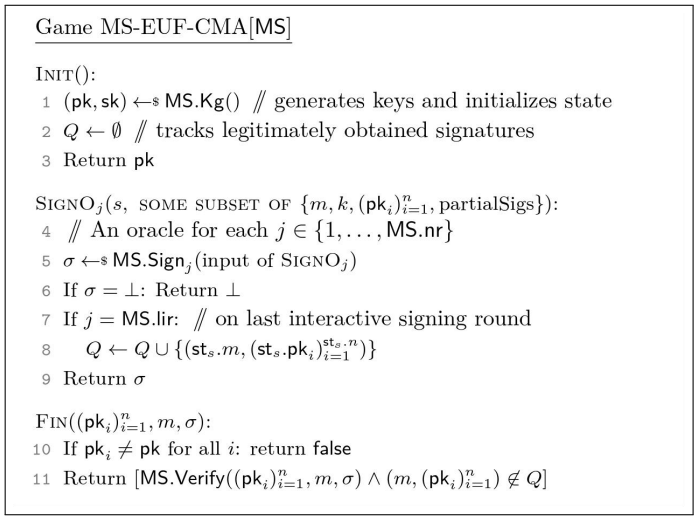

Figure 1: game used to define the existential unforgeability of a multi-signature scheme MS.

and signing group. However, this definition is insufficient, since the adversary should not be able to produce a signature unless the honest signer *completed* the signing protocol for the corresponding message and signing group. We conjecture that using this definition is one of the reasons that the referenced papers did not consider the security of BN and MuSig with delayed message selection. Consequently, in our syntax a scheme specifies its last interactive round MS.lir, and only at a signing oracle query for that round we record that a legitimate multi-signature has been provided.

A pseudocode game based definition of the MS-EUF-CMA game is presented in Figure 1.

Setting for our attacks. Our attacks against MuSig and BN with delayed message selection use a weaker adversary than that permitted by the existential unforgeability definition, which achieves more. In particular, an adversary can forge multi-signatures for a message of their choice without corrupting any of the signers in the group. See Section 4 for more details, but note that our attack breaks existential unforgeability regardless.

3 The Schemes

We now describe the multi-signature schemes that are the subject of this note. We ignore chronological order and describe MuSig first. This is for consistency with the attacks section (Section 4), which presents the simpler attacks first.

3.1 MuSig

We describe MuSig in words, as well as in pseudocode in Figure 2 which includes both the secure variant and the insecure version with delayed message selection.[3] We use the scheme as described by Bellare and Dai [BD21], which has minor differences from prior descriptions [MPSW19, BDN18]. This is for compatibility with our syntax and security definition, and the conclusions of this note stand for all variants.

[3]We only consider the revised three-round MuSig scheme and not the original two-round version [MPSW18], which is completely insecure. This is due to an irreparable bug in the security proof [DEF+19] and a later efficient attack [BLL+21] that involves solving the ROS problem [Sch01].

MuSig is parameterized by a group $\mathbb{G}$ of prime order p with a generator g, and three hash functions H_{com}, H_{sign}, and H_{agg} with codomain $\mathbb{Z}_p$ that are used for commitments, signing, and key aggregation respectively.

For key generation, each signer k generates a secret key $\mathsf{sk}_k \leftarrow\!\!\$\, \mathbb{Z}_p$ and a public key $\mathsf{pk}_k \leftarrow g^{\mathsf{sk}_k}$. The aggregate public key $\widetilde{\mathsf{pk}}$ of a group of n signers is

$$\widetilde{\mathsf{pk}} \leftarrow \prod_{i=1}^{n} \mathsf{pk}_i^{H_{\mathsf{agg}}(i,\mathsf{pk}_1,\ldots,\mathsf{pk}_n)}.$$

In the first signing rounds, each signer k chooses $r_k \leftarrow\!\!\$\, \mathbb{Z}_p$, computes $R_k \leftarrow g^{r_k}$, and outputs a commitment $t_k \leftarrow H_{\mathsf{com}}(R_k)$ that is sent to all the other signers. In the second round, the signer receives the commitments $t_1, \ldots, t_n$, and sends R_k to all other signers. In the third round, the signer receives nonces $R_1, \ldots, R_n$ and verifies the commitments by checking that $t_i = H_{\mathsf{com}}(R_i)$ for each i. Then, they compute $R \leftarrow \prod_{i=1}^{n} R_i$, the aggregate public key $\widetilde{\mathsf{pk}}$ as described before, and a challenge $c \leftarrow H_{\mathsf{sign}}(\widetilde{\mathsf{pk}}, R, m)$, and output a signature share $z_k \leftarrow r_k + \mathsf{sk}_k \cdot c \cdot H_{\mathsf{agg}}(k, \mathsf{pk}_1, \ldots, \mathsf{pk}_n)$. Now, any of the signer can output the multi-signature (R, z) where $R \leftarrow \prod_{i=1}^{n} R_i$ and $z \leftarrow \sum_{i=1}^{n} z_i$.

A multi-signature (R, z) is valid with respect to an aggregated verification key $\widetilde{\mathsf{pk}}$ and a message m if and only if

$$g^z = R \cdot \widetilde{\mathsf{pk}}^{H_{\mathsf{sign}}(\widetilde{\mathsf{pk}},R,m)}.$$

Correctness is easy to verify. Furthermore, the verification of a MuSig multi-signature with respect to an aggregated key $\widetilde{\mathsf{pk}}$ is identical to the verification of a standard Schnorr signature, adding to the appeal of the scheme.

Which signing rounds are message dependent. The signers in MuSig do not use the message in the first two signing rounds. Thus, it is natural to ask whether the message needs to be determined at the initiation of the protocol? Alternatively, is it possible to pre-process the first two signing rounds before the message is selected, and thus have only one remaining round when the message is determined? As mentioned in the introduction, prior literature is ambiguous regarding this question.

In Section 4 we show that the answer is no. The signers must associate each signing session with a message and a signing group before executing the second signing round (the "reveal" round of the nonce shares). In other words, the signers must store the message to be signed before the second round, even though it is not used until the third round, else the scheme is no longer unforgeable.

MuSig security proofs. MuSig is proven secure under the discrete log assumption when the hash functions are modeled as a random oracle [MPSW19, BDN18, BD21]. None of those papers consider whether MuSig can be used with delayed message selection.

3.2 Bellare-Neven Multi-Signatures

The Bellare-Neven scheme (abbreviated BN) [BN06] is a predecessor of MuSig and was the first secure scheme in the plain public key model, meaning the setting where the signers do not need to participate in a distributed key generation protocol or prove ownership of a secret key. It is very similar to MuSig but without key aggregation, and is also proven secure under the discrete log assumption when the message to sign is determined before the second signing round and the hash functions are modeled as random oracles. As with MuSig, we follow the scheme description of Bellare and Dai [BD21], which has minor differences from the original paper [BN06].

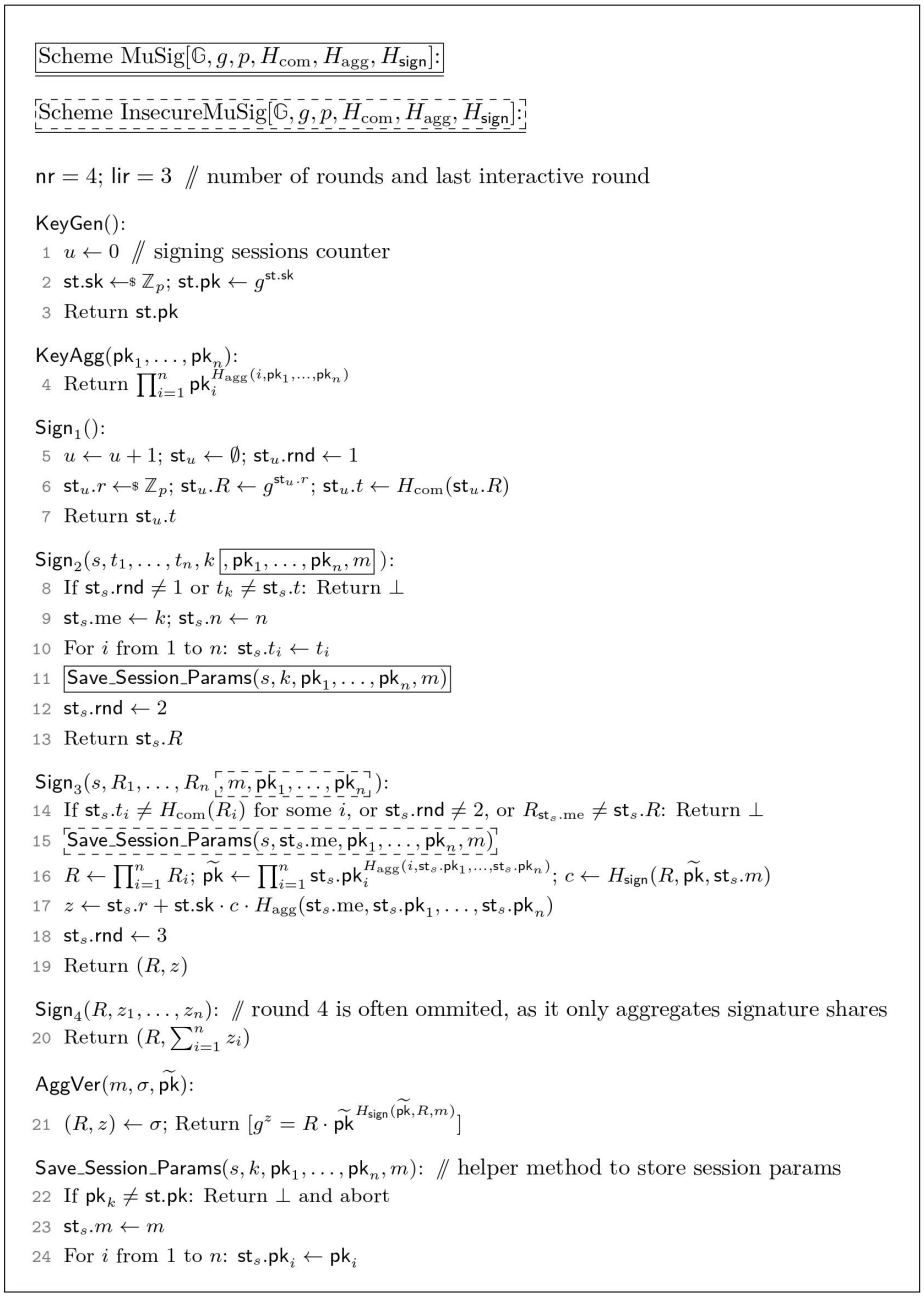

Figure 2: a description of MuSig using our syntax. The secure version contains all but the dashed boxes, and the insecure version with delayed message selections contains all but the solid boxes.

The scheme is parameterized by a group $\mathbb{G}$ of prime order p with a generator g and two hash functions H_{com} and H_{sign} with codomain $\mathbb{Z}_p$ that are used for commitments and signing respectively.

Key generation and the first two signing rounds were left unaltered by MuSig. For key generation, each signer k generates a private key $\mathsf{sk}_k \leftarrow\!\!\$\ \mathbb{Z}_p$ and a public key $\mathsf{pk}_k \leftarrow g^{\mathsf{sk}_k}$. For the first signing round, signer k chooses $r_k \leftarrow\!\!\$\ \mathbb{Z}_p$, computes $R_k \leftarrow g^{r_k}$, and outputs a commitment $t_k \leftarrow H_{\mathsf{com}}(R_k)$ which is sent to all the other signers. In the second signing round, the signer receives the commitments from all other signers $t_1, \ldots, t_n$, and outputs R_k. In the third signing round, which is different from MuSig, the signer k receives nonces $R_1, \ldots, R_n$ from all the signers and verifies the commitments by checking that $t_i = H_{\mathsf{com}}(R_i)$ for each i. Then, they compute $R \leftarrow \prod_{i=1}^{n} R_i$, a challenge $c_k \leftarrow H_{\mathsf{sign}}(k, R, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, m)$, and output a signature share $z_k \leftarrow r_k + \mathsf{sk}_k \cdot c_k$. Now, any of the signer can output the multi-signature (R, z) where $R \leftarrow \prod_{i=1}^{n} R_i$ and $z \leftarrow \sum_{i=1}^{n} z_i$.

Verification requires the message m, the signature $\sigma = (R, z)$, and all the signers public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$. The verifier computes $c_i \leftarrow H_{\mathsf{sign}}(i, R, \mathsf{pk}_1, \ldots, \mathsf{pk}_n, m)$ for each $i \in \{1, \ldots, n\}$ and output true if and only if $g^z = R \prod_{i=1}^{n} \mathsf{pk}_i^{c_i}$.

As with MuSig, the message and the keys of all the signers are not used until the third signing round. However, as our attack shows, it is needed for security that the message being signed is selected before the second signing round.

4 The Attacks

Here we present our attacks against MuSig and BN multi-signatures when used with delayed message selection. We first present a sub-exponential attack against each scheme using Wagner's algorithm for the generalized birthday algorithm [Wag02]. Then, we build on the ideas from the sub-exponential attacks and use the algorithm solving the ROS problem of Benhamouda et al. [BLL+21] to design a more efficient polynomial time attack in the same setting.

Suppose the first two rounds of MuSig and BN multi-signatures (the commitments and revealing the nonces rounds) are executed before message selection. A pseudocode description of this insecure version of MuSig and comparison with the secure version is provided in Figure 2. We will describe the attacks when executed with two signers, but it is straightforward to generalize it to a setting with more signers.

Attack setting. The standard existential unforgeability definition allows the adversary to corrupt all but one of the signers in a group, as well as ask the honest signer for signatures with differing groups. Furthermore, to break existential unforgeability the adversary only needs to forge a signature for an arbitrary message. All of our attacks can be carried out by a weaker adversary, who can forge a signature for a message of their choice.

In particular, in the attacks against MuSig the adversary only needs to observe parallel signing sessions and control which message will be signed, but can succeed against any group of signers even if none of them is corrupt. Our attacks against BN multi-signatures also don't require the adversary to collude with signers, but the adversary needs the honest signers to complete a signing session for different messages. This is possible when the adversary mediates signer communication or corrupts all but one of the signers.

4.1 Warm Up: Sub-Exponential Attacks using Wagner's Algorithm

In Crypto 2002, Wagner initiated the study of the generalized birthday problem, and presented a sub-exponential algorithm to solve it [Wag02]. The variation of the generalized

birthday problem that is useful to us is the following:

> **The generalized birthday problem.** Given $c \in \mathbb{Z}_p$ and ℓ lists (of arbitrary length) $L_1, \ldots, L_\ell$ of elements drawn uniformly and independently at random from $\mathbb{Z}_p$, find $c_1 \in L_1, \ldots, c_\ell \in L_\ell$ such that $\sum_{i=1}^{\ell} c_i = c$ (where the addition is mod p).

In the original paper [Wag02], Wagner conjectures that his algorithm solves this problem with non-negligible probability in $O(\ell \cdot p^{1/(1+\lfloor \log(\ell) \rfloor)})$, and later analysis shows that the conjecture is true [JKL24]. This results in sub-exponential runtime when ℓ is large, though not polynomial in $\log(p)$.

We present simple attacks against MuSig and BN multi-signatures with delayed message selection that uses Wagner's algorithm as a black box. For these attacks, we also assume that the hash functions used by the schemes behave like random oracles, though this assumption is not needed for the polynomial time attacks in Section 4.2.

The Attack Against MuSig

This attack against MuSig is due to Jonas Nick [Nic19], and is included here for the sake of completion.

Consider signers S_1 and S_2 with private keys x_1 and x_2 and public key X_1 and X_2, respectively. Let $\widetilde{X} = X_1^{H_{\mathsf{agg}}(1,X_1,X_2)} \cdot X_2^{H_{\mathsf{agg}}(2,X_1,X_2)}$ denote the aggregate verification key. Let ℓ be an integer that can be adjusted to optimize the runtime, and let m be the message for which the adversary wishes to forge a signature.

Now, the adversary begins ℓ signing sessions and observes the first two signing rounds to obtain an aggregate nonce $R_i = R_{i,1} \cdot R_{i,2}$ for each $i \in \{1, \ldots, \ell\}$. The adversary calculates $R = \prod_{i=1}^{\ell} R_i$ and a challenge $c = H_{\mathsf{sign}}(\widetilde{X}, R, m)$. Now, define ℓ lists $L_1, \ldots, L_\ell$ such that $L_{i,j} = H_{\mathsf{sign}}(\widetilde{X}, R_i, m_j)$ where the m_j are some arbitrary messages that are distinct for different j.

The adversary can now use Wagner's algorithm to find elements $c_i = L_{i,j_i} \in L_i$ for each list such that $\sum_{i=1}^{\ell} c_i = c$ (where, as always, summation is mod p). Next, the adversary finishes each signing session i to sign the message m_{j_i}, obtaining a valid multi-signature $\sigma_i = (R_i, z_i)$ for the message m_{j_i}. Now, the adversary outputs the forged signature $\sigma = (R, \sum_{i=1}^{\ell} z_i)$.

Validity of forged signature. We wish to verify that $\sigma = (R, \sum_{i=1}^{\ell} z_i)$ is a valid signature for the message m under the aggregate verification key $\widetilde{X}$. Equivalently, we must show that

$$g^{\sum_{i=1}^{\ell} z_i} = R\widetilde{X}^c.$$

Note that for each $i \in \{1, \ldots, \ell\}$, the signature $\sigma_i = (R_i, z_i)$ is valid for the message m_{j_i}, and thus $g^{z_i} = R_i \cdot \widetilde{X}^{c_i}$. Therefore,

$$\prod_{i=1}^{\ell} g^{z_i} = \prod_{i=1}^{\ell} R_i \widetilde{X}^{c_i},$$

or equivalently,

$$g^{\sum_{i=1}^{\ell} z_i} = R\widetilde{X}^{\sum_{i=1}^{\ell} c_i}.$$

However, by construction $\sum_{i=1}^{\ell} c_i = c$, and therefore we conclude that $g^{\sum_{i=1}^{\ell} z_i} = R\widetilde{X}^c$, which is what we wanted to prove.

The Attack Against BN Multi-Signatures

As before, let S_1 and S_2 be the signers with private keys x_1 and x_2 and public key X_1 and X_2, respectively. Let ℓ be an integer that can be adjusted to optimize the runtime, and let m be the message for which the adversary wishes to forge a signature.

The adversary begins ℓ signing sessions and observes the first two signing rounds to obtain nonce shares $R_{i,1} = g^{r_{i,1}}$ and $R_{i,2} = g^{r_{i,2}}$ and the aggregate nonce $R_i = R_{i,1}R_{i,2}$ for each $i \in \{1, \ldots, \ell\}$. The adversary computes $R = \prod_{i=1}^{\ell} R_i$ and challenges $c_1 = H_{\mathsf{sign}}(1, R, X_1, X_2, m)$ and $c_2 = H_{\mathsf{sign}}(2, R, X_1, X_2, m)$.

The adversary now attempts to forge a signature share for S_1 with the message m, aggregate nonce R, and nonce share $\prod_{i=1}^{n} R_{i,1}$. It defines ℓ lists $L_1, \ldots, L_\ell$ such that $L_{i,j} = H_{\mathsf{sign}}(1, R_i, X_1, X_2, m_j)$, where the m_j are arbitrary messages that are distinct for different j. Using Wagner's algorithm, the adversary finds elements $c_{i,1} = L_{i,j_i} \in L_i$ for each list, such that $\sum_{i=1}^{\ell} c_{i,1} = c_1$. Now, the adversary asks S_1 to finish each signing session i with the message m_{j_i}, obtaining a signature share $z_{i,1} = r_{i,1} + c_{i,1}x_1$. Finally, they output the forged signature share $z_1 = \sum_{i=1}^{\ell} z_{i,1}$.

To break existential unforgeability, it is sufficient to forge a multi-signature in the case where S_2 colludes with the adversary. In this case, the adversary knows the state of S_2 and can compute $z_2 = \sum_{i=1}^{\ell} r_{i,2} + x_2 c_2$, which is a valid signature share for S_2 with the message m, nonce R, and nonce share $\prod_{i=1}^{\ell} R_{i,2}$. Otherwise, if S_2 is honest, the adversary can follow the same procedure as done for S_1 to forge the signature share z_2 with the same nonce and nonce share. Note that this requires S_2 to finish the opened signing sessions with different messages than S_1, requiring the adversary to control their communications. Given the two signature shares, the adversary can forge the final multi-signature $\sigma = (R, z_1 + z_2)$.

Validity of forged signature. The validity of the forged multi-signature follows from the validity of each signature share. In other words, if $z_1 = \sum_{i=1}^{\ell} r_{i,1} + x_1 \cdot c_1$ and $z_2 = \sum_{i=1}^{\ell} r_{i,2} + x_2 \cdot c_2$, then σ is a valid forgery.

First, note that

$$z_1 = \sum_{i=1}^{\ell} z_{i,1} = \sum_{i=1}^{\ell} r_{i,1} + x_1 \sum_{i=1}^{\ell} c_{i,1}.$$

However, by construction (using Wagner's algorithm) $\sum_{i=1}^{\ell} c_{i,1} = c_1$, which means that $z_1 = \sum_{i=1}^{\ell} r_{i,1} + x_1 \cdot c_1$, and therefore z_1 is a valid signature share. The fact that z_2 is also valid follow trivially from its construction in the case where S_2 is corrupt, and in the case where S_2 is honest its validity can be seen using the same reasoning as the validity of z_1.

4.2 Polynomial Time Attacks

The algorithm for these attacks is a variation of the algorithm for solving the ROS problem [BLL+21], which broke the original two-round variant of MuSig [MPSW18] among many other multi-, threshold, and blind signature schemes.

The Attack Against MuSig

As usual, let S_1 and S_2 be the signers with private keys x_1 and x_2 and public key X_1 and X_2, respectively. Let $\widetilde{X} = X_1^{H_{\mathsf{agg}}(1,X_1,X_2)} \cdot X_2^{H_{\mathsf{agg}}(2,X_1,X_2)}$ denote the aggregate verification key. This time, let $\ell \geq \lceil \log_2(p) \rceil$ be an integer and let $m_{\ell+1}$ be some message for which the adversary wishes to forge a signature, and for each $i \in \{1, \ldots, \ell\}$ choose distinct messages m_i^0 and m_i^1 that the signers would be willing to sign.

Now, the adversary begins ℓ signing sessions and observes the first two signing rounds to obtain an aggregate nonce $R_i = R_{i,1} \cdot R_{i,2}$ for each $i \in \{1, \ldots, \ell\}$. Then the adversary

calculates the corresponding challenges $c_i^0 = H_{\mathsf{sign}}(R_i, \widetilde{X}, m_i^0)$ and $c_i^1 = H_{\mathsf{sign}}(R_i, \widetilde{X}, m_i^1)$ for each $i \in \{1, \ldots, \ell\}$. Now, define the group homomorphisms $\rho^+ : (\mathbb{Z}_p)^\ell \to \mathbb{Z}_p$ and $\rho^\times : (\mathbb{G})^\ell \to \mathbb{G}$ as follows:

$$\rho^+(x_1, \ldots, x_\ell) = \sum_{i=1}^{\ell} \frac{2^{i-1} x_i}{c_i^1 - c_i^0},$$

and

$$\rho^\times(g_1, \ldots, g_\ell) = \prod_{i=1}^{\ell} g_i^{\frac{2^{i-1}}{c_i^1 - c_i^0}}.$$

Let $R_{\ell+1} = \rho^\times(R_1, \ldots, R_\ell)$, and let $c_{\ell+1} = H_{\mathsf{sign}}(R_{\ell+1}, \widetilde{X}, m_{\ell+1})$. Let $d = c_{\ell+1} - \rho^+(c_1^0, \ldots, c_\ell^0)$, and write d in binary as $\sum_{i=1}^{\ell} 2^{i-1} b_i$ for some $b_1, \ldots, b_\ell \in \{0, 1\}$, which is possible since $\ell \geq \lceil \log_2(p) \rceil$. Now, for each $i \in \{1, \ldots, \ell\}$, complete the signing session i with the message $m_i^{b_i}$ to obtain a multi-signature (R_i, z_i). We claim that $\sigma = (R_{\ell+1}, \rho^+(z_1, \ldots, z_\ell))$ is a valid multi-signature for the message $m_{\ell+1}$ under the aggregate verification key $\widetilde{X}$, and is thus a forgery that breaks the unforgeability of the scheme.

Validity of forged signature. We wish to verify that $\sigma = (R_{\ell+1}, \rho^+(z_1, \ldots, z_\ell))$ is a valid signature for $m_{\ell+1}$ under the aggregate verification key $\widetilde{X}$. Thus, we must show that

$$g^{\rho^+(z_1, \ldots, z_\ell)} = R_{\ell+1} \widetilde{X}^{c_{\ell+1}}.$$

Note that (R_i, z_i) is a valid Schnorr signature for the message $m_i^{b_i}$ under the verification key $\widetilde{X}$ for each i, and thus $g^{z_i} = R_i \widetilde{X}^{c_i^{b_i}}$. Hence,

$$\rho^\times(g^{z_1}, \ldots, g^{z_\ell}) = \rho^\times(R_1 \widetilde{X}^{c_1^{b_1}}, \ldots, R_\ell \widetilde{X}^{c_\ell^{b_\ell}}),$$

or equivalently,

$$g^{\rho^+(z_1, \ldots, z_\ell)} = \rho^\times(R_1, \ldots, R_\ell) \cdot \widetilde{X}^{\rho^+(c_1^{b_1}, \ldots, c_\ell^{b_\ell})}.$$

But $R_{\ell+1} = \rho^\times(R_1, \ldots, R_\ell)$, and Lemma 1 shows that $\rho^+(c_1^{b_1}, \ldots, c_\ell^{b_\ell}) = c_{\ell+1}$. Hence,

$$g^{\rho^+(z_1, \ldots, z_\ell)} = R_{\ell+1} \cdot \widetilde{X}^{c_{\ell+1}},$$

which is what we wanted to prove.

Lemma 1. *By the construction above,* $\rho^+(c_1^{b_1}, \ldots, c_\ell^{b_\ell}) = c_{\ell+1}$.

This lemma is at the heart of the attack, and is precisely the idea that allows Benhamouda et al. to solve the ROS problem [BLL+21].

Proof of Lemma 1. By definition, $\sum_{i=1}^{\ell} 2^{i-1} b_i = c_{\ell-1} - \rho^+(c_1^0, \ldots, c_\ell^0)$. Hence, to prove the lemma it is sufficient to show that $\sum_{i=1}^{\ell} 2^{i-1} b_i = \rho^+(c_1^{b_1}, \ldots, c_\ell^{b_\ell}) - \rho^+(c_1^0, \ldots, c_\ell^0)$.

Starting from the right-hand side, we have that

$$\begin{aligned} \rho^+(c_1^{b_1}, \ldots, c_\ell^{b_\ell}) - \rho^+(c_1^0, \ldots, c_\ell^0) &= \rho^+(c_1^{b_1} - c_1^0, \ldots, c_\ell^{b_\ell} - c_\ell^0) \\ &= \sum_{i=1}^{\ell} \frac{2^{i-1}(c_i^{b_i} - c_i^0)}{c_i^1 - c_i^0}. \end{aligned}$$

However, for each i it holds that $\frac{2^{i-1}(c_i^{b_i} - c_i^0)}{c_i^1 - c_i^0}$ is 0 whenever b_i is 0 and is 2^{i-1} whenever b_i is 1. Hence, for each i it holds that $\frac{2^{i-1}(c_i^{b_i} - c_i^0)}{c_i^1 - c_i^0} = 2^{i-1} b_i$, and thus the right-hand side of the equation above simplifies to $\sum_{i=1}^{\ell} 2^{i-1} b_i$, which completes the proof.

The Attack Against BN Multi-Signatures

In contrast to MuSig, different signers in BN multi-signatures use distinct challenges when signing a message. At a high level, we now need a separate instance of the ROS attack that we used against MuSig for each signer in order to forge their partial signature.

As always, let S_1 and S_2 be the signers with private keys x_1 and x_2 and public keys X_1 and X_2, respectively. Let $\ell \geq \lceil \log_2(p) \rceil$ be an integer, let $m_{\ell+1}$ be some message for which the adversary wishes to forge a multi-signature, and for each $i \in \{1, \ldots, \ell\}$ choose distinct messages m_i^0 and m_i^1 that the signers would be willing to sign.

Now, the adversary begins ℓ signing sessions and observes the first two signing rounds to obtain nonce shares $R_{i,1} = g^{r_{i,1}}$ and $R_{i,2} = g^{r_{i,2}}$ for each $i \in \{1, \ldots, \ell\}$. Then, the adversary calculates challenges $c_{i,1}^b = H_{\mathsf{sign}}(1, R_{i,1} \cdot R_{i,2}, X_1, X_2, m_i^b)$ and $c_{i,2}^b = H_{\mathsf{sign}}(2, R_{i,1} \cdot R_{i,2}, X_1, X_2, m_i^b)$ for each $i \in \{1, \ldots, \ell\}$ and $b \in \{0, 1\}$. Now, define the group homomorphisms $\rho_j^+ : (\mathbb{Z}_p)^\ell \to \mathbb{Z}_p$ and $\rho_j^\times : (\mathbb{G})^\ell \to \mathbb{G}$ for $j \in \{1, 2\}$ as follows:

$$\rho_j^+(x_1, \ldots, x_\ell) = \sum_{i=1}^{\ell} \frac{2^{i-1} x_i}{c_{i,j}^1 - c_{i,j}^0},$$

and

$$\rho_j^\times(g_1, \ldots, g_\ell) = \prod_{i=1}^{\ell} g_i^{\frac{2^{i-1}}{c_{i,j}^1 - c_{i,j}^0}}.$$

Let $R_1 = \rho_1^\times(R_{1,1}, \ldots, R_{\ell,1})$ and $R_2 = \rho_2^\times(R_{1,2}, \ldots, R_{\ell,2})$. Let $R = R_1 \cdot R_2$, let $c_{\ell+1,1} = H_{\mathsf{sign}}(1, R, X_1, X_2, m_{\ell+1})$, and also let $c_{\ell+1,2} = H_{\mathsf{sign}}(2, R, X_1, X_2, m_{\ell+1})$. Let $d_1 = c_{\ell+1,1} - \rho_1^+(c_{1,1}^0, \ldots, c_{\ell,1}^0)$ and write it in binary as $\sum_{i=1}^{\ell} 2^{i-1} b_{i,1}$ for some $b_{1,1}, \ldots, b_{\ell,1} \in \{0, 1\}$, which is possible since $\ell \geq \lceil \log_2(p) \rceil$. Similarly, let $d_2 = c_{\ell+1,2} - \rho_2^+(c_{1,2}^0, \ldots, c_{\ell,2}^0)$ and write it in binary as $\sum_{i=1}^{\ell} 2^{i-1} b_{i,2}$.

Now, for each $i \in \{1, \ldots, \ell\}$ complete the third signing round of the signing session i with S_1 using the message $m_i^{b_{i,1}}$ to obtain a signature share $z_{i,1} = r_{i,1} + c_{i,1}^{b_{i,1}} \cdot x_1$. Similarly, complete the third signing round of session i with S_2 using the (potentially different) message $m_i^{b_{i,2}}$ to obtain a signature share $z_{i,2} = r_{i,2} + c_{i,2}^{b_{i,2}} \cdot x_2$. Now, they can calculate $z_{\ell+1,1} = \rho_1^+(z_{1,1}, \ldots, z_{\ell,1})$ and $z_{\ell+1,2} = \rho_2^+(z_{1,2}, \ldots, z_{\ell,2})$.

We claim that $\sigma = (R, z_{\ell+1,1} + z_{\ell+1,2})$ is a valid multi-signature for the message $m_{\ell+1}$ under the group S_1 and S_2, and thus this attack breaks the existential unforgeability of the scheme.

Validity of forged signature. We wish to verify that $\sigma = (R, z_{\ell+1,1} + z_{\ell+1,2})$ is a valid multi-signature for the message $m_{\ell+1}$ and the group of signers S_1 and S_2. Hence, we must show that

$$g^{z_{\ell+1,1} + z_{\ell+1,2}} = R \cdot X_1^{c_{\ell+1,1}} \cdot X_2^{c_{\ell+1,2}}.$$

Starting from the left-hand side, we have that

$$\begin{aligned} g^{z_{\ell+1,1} + z_{\ell+1,2}} &= g^{\rho_1^+(z_{1,1}, \ldots, z_{\ell,1}) + \rho_2^+(z_{1,2}, \ldots, z_{\ell,2})} \\ &= \rho_1^\times(R_{1,1}, \ldots, R_{\ell,1}) \cdot \rho_2^\times(R_{1,2}, \ldots, R_{\ell,2}) \cdot X_1^{\rho_1^+(c_{1,1}^{b_{1,1}}, \ldots, c_{\ell,1}^{b_{\ell,1}})} X_2^{\rho_2^+(c_{1,2}^{b_{1,2}}, \ldots, c_{\ell,2}^{b_{\ell,2}})}. \end{aligned}$$

The first two terms in the above equation are R_1 and R_2. Furthermore, $\rho_1^+(c_{1,1}^{b_{1,1}}, \ldots, c_{\ell,1}^{b_{\ell,1}}) = c_{\ell+1,2}$ and $\rho_2^+(c_{1,2}^{b_{1,2}}, \ldots, c_{\ell,2}^{b_{\ell,2}}) = c_{\ell+1,2}$ using the same idea as in Lemma 1. Thus, the above equation simplifies to

$$g^{z_{\ell+1,1} + z_{\ell+1,2}} = R \cdot X_1^{c_{\ell+1,1}} \cdot X_2^{c_{\ell+1,2}},$$

which is what we wanted to prove.

Acknowledgements

I thank Stefano Tessaro for the discussions leading to this note, as well as his invaluable assistance in its writing and revision.

References

[AB21] Handan Kilinç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 157–188, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84242-0_7.

[BD21] Mihir Bellare and Wei Dai. Chain reductions for multi-signatures and the HBMS scheme. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 650–678, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-92068-5_22.

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 435–464, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-03329-3_15.

[BLL+21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 33–53, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-77870-5_2.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 390–399, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. doi:10.1145/1180405.1180453.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. doi:10.1007/11761679_25.

[BTT22] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 276–305, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-15979-4_10.

[CKM21] Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. Cryptology

ePrint Archive, Paper 2021/1375, 2021. URL: https://eprint.iacr.org/2021/1375.

[DEF+19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101, San Francisco, CA, USA, May 19–23, 2019. IEEE Computer Society Press. doi:10.1109/SP.2019.00050.

[IN83] K Itakura and K Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC research & development*, 1983.

[JKL24] Antoine Joux, Hunter Kippen, and Julian Loss. A concrete analysis of wagner's k-list algorithm over $\mathbb{Z}_p$. Cryptology ePrint Archive, Paper 2024/282, 2024. URL: https://eprint.iacr.org/2024/282.

[MPSW18] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin (deprecated version). Cryptology ePrint Archive, Report 2018/068, version 1, 2018. URL: https://eprint.iacr.org/archive/2018/068/20180118:124757.

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Designs, Codes and Cryptography*, 87(9):2139–2164, Sep 2019. doi:10.1007/s10623-019-00608-x.

[Nic19] Jonas Nick. Insecure shortcuts in musig, 2019. URL: https://medium.com/blockstream/insecure-shortcuts-in-musig-2ad0d38a97da.

[NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 189–221, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84242-0_8.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. doi:10.1007/0-387-34805-0_22.

[Sch01] Claus Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *Information and Communications Security*, pages 1–12, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. doi:10.1007/3-540-45600-7_1.

[TZ23] Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 628–658, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-30589-4_22.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. doi:10.1007/3-540-45708-9_19.